Constructing complex minds through multiple authors

Mark Humphrys[1,3]
[1]School of Computer Applications,
Dublin City University,
Glasnevin, Dublin 9, Ireland
`www.compapp.dcu.ie/~humphrys`

Ciarán O'Leary[2,3]
[2]School of Computing,
Dublin Institute of Technology,
Kevin St, Dublin 8, Ireland
`comp.dit.ie/coleary`

[3]The World-Wide-Mind project
`w2mind.org`

Abstract

The World-Wide-Mind (WWM) was introduced in [Humphrys, 2001]. For a short introduction see [Humphrys, 2001a]. Briefly, this is a scheme for putting animat "minds" online (as WWM "servers") so that large complex minds may be constructed from many remote components. The aim is to address the *scaling up* of animat research, or how to construct minds more complex than could be written by one author (or one research group).

The first part of this paper describes how a number of existing animat architectures could be implemented as WWM servers. Any *unified* mind can easily map to a *single* WWM server. So most of the discussion here is on *action selection* (or behavior or goal selection), where each module could be a different WWM server (written by a different author).

The second part of this paper describes the first implementation of WWM servers and clients, and explains in particular how to write a WWM server. Most animats researchers are programmers but not network programmers. Almost all protocols for remote services (CORBA, SOAP, etc.) assume the programmer is a networks specialist. This paper rejects these solutions, and shows how any animats researcher can put their animat "mind" or "world" online as a server by simply converting it into a command-line program that reads standard input and writes to standard output.

1 Introduction

1.1 The AI problem

"AI" refers here to all artificial modelling of life, animals and humans. In the sense in which we use it, classic symbolic AI, sub-symbolic AI, Animats, Agents and ALife are all subfields of "AI".

It is generally agreed that the AI problem is much harder than researchers used to think (though it is not clearly understood *why*). Early optimism has given way to a sober respect for the magnitude of the problem, and a number of approaches have evolved:

1. The standard AI approach has been to work on *subsections* of the postulated mind, such as computer vision, or language processing. The criticism of this approach [Brooks, 1986, Brooks, 1991] is that the whole mind never actually gets built [Nilsson, 1995].

2. The *Animats* approach [Wilson, 1990] is to start with *simple whole creatures* [Dennett, 1978] and work up gradually to more complex whole creatures.

3. The *evolutionary* approach is to say that control systems are too hard to design and must be evolved [Harvey et al., 1992]. In practice this has usually taken the animat approach of starting with simple whole creatures.

It may be time to ask questions about how the animats and evolutionary approaches scale up. Both seem to share an implicit assumption that *one lab can do it all.* As a result, the complexity of the minds produced is limited to the complexity that can be grasped by a single research team (or even a single individual). Perhaps the Cog project [Brooks, 1997, Brooks et al., 1998] is beginning to hit the limits of what a single coherent team can understand.

1.2 Constructing complex minds through multiple authors

What is the alternative? The alternative is to *link the work of multiple laboratories* - to construct minds out of sub-minds written by many diverse authors. This is done of course already within research groups, but we propose a public system, where perhaps hundreds of diverse authors may contribute parts to a large, complex mind. Other researchers may specialise *entirely* on just finding different ways of combining other people's components. No one individual need understand the entire mind.

This is, of course, the problem of *building whole minds out of specialist components* that standard AI never solved, and we suggest why. We argue that only by using a public, open system on the Internet as the infrastructure on which to build the mind can this problem be solved. We do not suggest the abolition of the animats approach, but rather modifying it to building *simple whole creatures* out of components written by multiple authors, and scale up to building *complex whole creatures* out of components written by multiple authors.

1.3 "Not enough stuff" in AI

AI has a history of people saying that the systems simply need *more* of some property and there will be a breakthrough:

1. **Not enough speed.** - "If we just get faster machines we will have full AI". This thinking is common when AI is viewed primarily as a *search* problem - and there is obviously a lot of truth in it (see AI's recent triumph at chess [McDermott, 1997]). This thinking has also been revived with the discussion of hypothetical super-powerful quantum computers that might be able to solve NP-complete problems in polynomial time. Some writing on this (e.g. [Berger, 1998]) suggests that if such super-powerful computers can be made, then AI will be solved.

2. **Not enough neurons/memory.** - "Brain-building" [de Garis, 1996] will, it is claimed, lead to breakthroughs once enough neurons are put together.

3. **Not enough data.** - The CYC project [cyc.com], and the online projects MindPixel [mindpixel.com] and Open Mind [openmind.org], take the approach that what is needed is to build up vast rule sets.

Here we suggest another one:

1. **Not enough authors.** - Not enough diversity in the mind.

The criticism of "not enough stuff" arguments is that *theoretical* breakthroughs are needed. However, if you look at the brain, it *is* true that it has more "stuff" - i.e. neurons, connections, memories, experience, multiple learning algorithms, multiple specialised structures - than artificial models. The brain is much bigger and richer than any machine we have built so far (perhaps even bigger and richer than *all* of our machines plus the Internet). So the "not enough stuff" argument must be true on one level (while not disagreeing with the idea that it is *also* true that we will need theoretical breakthroughs).

The "not enough authors" approach is the one that has not yet been *tried*. Many researchers have emphasised the vast and *heterogenous* nature of the mind, notably Minsky [Minsky, 1986, Minsky, 1991]. In the Animats world it is at least accepted that complex minds will have "Action Selection" among competing sub-minds. So far, the case has been made for heterogenous minds, but no one has shown how to build *really* heterogenous minds.

2 The World-Wide-Mind (WWM)

The World-Wide-Mind (WWM) was introduced in [Humphrys, 2001]. For a short introduction see [Humphrys, 2001a]. Briefly, it is proposed that researchers construct their animat "minds" and "worlds" as *servers* on the Internet. Each WWM "server" is a program residing on a normal Web server. There are two basic types of server:

1. A **World** and Body server. This server can be queried for the current state of the world `x` as detected by the body, and can be sent actions `a` for the body to perform in the world. For example, one could put Tyrrell's action-selection world [Tyrrell, 1993] online as a server, and then people could write minds to drive the agent in it.

2. A **Mind** server, which is a behavior-producing system, capable of suggesting an action `a` given a particular input state `x`. This does not mean it is stimulus-response. It may remember *all* previous states. It may take actions independent of the current state. It may work by any AI methodology or algorithm and may contain within itself any degree of complexity.

A *user* "runs" a Mind server in a World server, using some dedicated *client software*. The user is typically remote from both Mind and World, and starts the client by giving it the (remote) World URL and Mind URL. The client then (repeatedly) queries the World for state, passes this to the Mind to get a suggested action, sends this to the World for execution, and so on.

2.1 How to construct a Society of Mind

Even in its basic form above, the scheme would allow remote re-use of other people's minds and worlds, something for which there is no easy scheme at present. But we may also consider using someone else's mind as merely a *component* in a larger mind. At the top level, there is always exactly one Mind server running in exactly one World server. But that Mind server may *itself* be calling many other Mind servers in order to select its action. We define the following types of Mind servers:

1. $Mind_M$ - a Mind server that calls other Mind servers.

2. $Mind_{AS}$ (or Action Selection or AS server) - a $Mind_M$ server that resolves competition among multiple Mind servers. Each Mind server `i` suggests an action a_i to execute. The AS server queries them and somehow produces a winning action a_k. To the outside world, the AS server looks like just another Mind server, producing action `a` given state `x`.

3. $Mind_i$ - a Mind server that accepts it may not be the only mind in the body (and may support additional queries to *cooperate* with the Action Selection).

4. $Mind_{Feu}$ - a Mind server that accepts *Feudal* commands of the form: "Take me to state c", rather than just commands of the form "What do you want to do now?".

5. $Mind_L$ - a Mind server that learns (and may support additional queries turning on and off learning).

Societies of mind may be incrementally built up using the servers that are online, for example:

1. 1st party makes World.
2. 2nd party makes Mind for World.
3. 3rd party makes $Mind_M$ which in state `x` does something, otherwise does what 2nd party Mind does.
4. 4th party makes different Mind for World.
5. 5th party makes $Mind_M$ which in state `y` does what 4th party Mind does otherwise does what 3rd party $Mind_M$ does. And so on, with people modifying and cautiously overriding what already works.

2.2 *What is the definition of* `x` *and* `a`*?*

Clearly, we *cannot* define a universal format for state `x` or action `a` that works across *all* worlds. We have no choice but to allow each World server define its own format of what the agent sees [e.g. gridworld state, 3D world description, neural network input vector, robotic sensory inputs] and what actions it can execute. Each Mind server that wants to run in that World will have to understand the format used.

What may prevent complete chaos, however, is first, that popular worlds will serve as benchmarks for testing. "Islands" of compatible worlds and minds may develop around each popular basic problem. Also, we *can* define a *client* that will work with all World servers and Mind servers. And finally, it *may* be the case that Minds can be written that will run in all worlds, or at least in a lot of quite different worlds. For example, one could write a *generic* Q-learning Mind server [Watkins, 1989]. When set to run in a new World server, it queries the world to learn that it has a finite number of states, numbered state `1` to state `n`, and a finite number of actions, action `1` to action `m`, and the World server will occasionally generate a numeric reward after an action has been taken. The Q-learning Mind server can then attempt to learn a policy without knowing anything more about what the world represents or what the problem is.

2.3 *Testing and methodological issues in AI*

Once systems are brought online in an open, public way, we can see that this could address some general methodological issues in the animats field (and other AI fields):

1. **Lack of re-use** - Sharing work has been so difficult that researchers tend to build their own animat minds and worlds from scratch, often duplicating work that has been done elsewhere. There have been a number of attempts to re-use animat or agent minds [Sloman and Logan, 1999, Sutton and Santamaria] and worlds [Daniels, 1999], but the model of re-use often requires installation, or even a particular programming language.
2. **Taking results on trust** - Often, the only person who *ever* does experiments with an animat or agent is its author. In this field it has become acceptable not to have direct access to many of the major systems under discussion. How many action selection researchers have ever *seen* Tyrrell's world running [Tyrrell, 1993] for example? This lack of direct experience is even greater when it comes to other researchers' *robotic* projects. We accept that we will never experiment with many of the systems under discussion ourselves, but only read papers on them.
3. **Lack of re-testing** - Lack of re-use has serious consequences for *scientific progress* in the sense of being able to *repeat* experiments and being able to *prove* that one system is better than another. [Bryson, 2000] points out that, essentially, no one uses each other's agent architectures, because they are not convinced by each other's tests. [Guillot and Meyer, 2000] make the same point about the animats field - that the number of architectures has grown faster than the number of comparisons. Having systems publicly available for indefinite re-testing by 3rd parties is, we argue, the only solution to this.

3 How to express animat architectures as networks of WWM servers

We now discuss how a number of existing animat architectures might be expressed as networks of WWM servers using simple remote queries. An attempt to define the full set of WWM server queries is in [Humphrys, 2001].

3.1 *A single (sub-symbolic or symbolic) Mind server*

A hand-coded mind program can clearly be implemented as a single Mind server, receiving `x` and returning `a`.

There are a vast number of models of animat or agent mind, whether hand-coded, learnt or evolved, symbolic or non-symbolic, that could be implemented as a single WWM server without raising any particular issues.

3.2 Multiple sub-symbolic Mind servers

The difficulty arises when we consider *competition* between multiple Minds. Any *unified* mind (such as a single learner trained to solve a task) can easily map to a *single* WWM server. So most of the discussion here is on *action selection* (or behavior or goal selection) among competing modules, where each module could be a different remote WWM server (written by a different remote author). Many of the Action Selection methods discussed below are surveyed in detail in [Humphrys, 1997], or, for a brief introduction, see [Humphrys, 1996].

3.3 This is easier to do at sub-symbolic level than at symbolic level

We concentrate initially on defining *sub-symbolic* level queries, where, for example, competition is resolved using numeric weights rather than by symbolic-level negotiation. This avoids the problem of symbolic *knowledge representation schemes* [Ginsberg, 1991] and *agent communication languages* [Martin et al., 2000] that has been the graveyard of so many previous attempts. Obviously the symbolic level will have to be addressed at *some* point, but we show below how much can be done before we get to that level, and how the WWM is currently more suitable for the *sub-symbolic minds* popular in the Animats, ALife and Neural Networks fields.

3.4 The Subsumption Architecture

A Subsumption Architecture model [Brooks, 1986, Brooks, 1991] could be implemented as a hierarchy of $Mind_M$ servers, each one building on the ones below it. Each one sends the current state `x` to the server below it, and then either uses their output or overrides it. As in Brooks' model, a set of lower layers will still work if the higher layers are removed. On the WWM, there may be many choices for (remote, 3rd party) higher layers to add to a given collection of lower layers.

3.5 Reinforcement Learning

An ordinary Reinforcement Learning (RL) agent, which receives rewards and punishments as it acts [Kaelbling et al., 1996], can clearly be implemented as a single Mind server. For example a Q-learning agent [Watkins, 1989] builds up Q-values ("Quality"-values) of how good each action is in each state: `Q(x,a)`. When learning, it can calculate a reward based on `x`, `a` and the new state `y`. So, so long as the client informs this Mind server what state `y` resulted from the previous action `a`, it can calculate rewards, and learn.

3.6 Hierarchical Q-Learning

Hierarchical Q-Learning [Lin, 1993] is a way of driving multiple Q-learners with a master Q-learner. It can be implemented on the WWM as follows. The client talks to a single $Mind_{AS}$ server, sending it `x` and receiving `a`. The $Mind_{AS}$ server talks to a number of Mind servers. The $Mind_{AS}$ server maintains a table of values `Q(x,i)` where `i` is which Mind server to pick in state `x`. Initially its choices are random, but by its own reward function, the $Mind_{AS}$ server fills in values for `Q(x,i)`. Having chosen `i`, it passes on the action suggested by Mind server `i` to the client. To save on the number of server queries (which is a more serious issue on the WWM than in a self-contained system), the $Mind_{AS}$ server does not query *any* of the Mind servers until it has picked an action `i`, and then it only queries a *single* Mind server `i`. There are a number of interesting possibilities:

1. The $Mind_{AS}$ server *need not know its list of Mind servers in advance.* It can be passed this list by the client at startup.
2. The subsidiary Mind servers *need not be Q-learners.* They could be any type of Mind server (including *symbolic* Mind servers), and the $Mind_{AS}$ server simply learns which one to let through.
3. A further possibility [thanks to Dave O'Connor] is that the Hierarchical Q-Learner could build `Q(x,i)` values for every Mind server on the Net. It acts as a spider, finding *new*, random Mind servers, and trying their actions out in pursuit of its goal. The result of all of this learning will be the construction of a huge map telling it which server `i` to pick in each state `x`. It may use hundreds of servers to implement its goal.

Because the number of queries made is a more important issue on a network system than on a *self-contained* system, we distinguish different types of $Mind_{AS}$ server:

1. AS_s **server** - makes a single query of each Mind server before making its decision.
2. AS_o **server** - does not even query *all* Mind servers *once.* It just makes one query of one Mind server.
3. AS_m **server** - makes multiple queries of each Mind server before making its decision.

Hierarchical Q-Learning is an AS_o server.

3.7 W-learning

We consider a number of schemes where Mind servers promote their actions with a weight W, or "W-value" [Humphrys, 1997]. Ideally the W-value will depend on the state `x` and will be higher or lower depending on how much the Mind server "cares" about winning the competition for this state. A *static* measure of the W-value is one in which the Mind server promotes its action with a value of W independent of the competition (e.g. W=Q). Any such method can clearly be implemented as a $Mind_i$ server. A *dynamic* measure of W is one in which the value of W changes depending on whether the Mind server was obeyed, and on what happened if it was not obeyed. Clearly this is an AS_s server that queries once, lets through the highest W, and then *reports back* afterwards to each Mind server whether or not it was obeyed. The server may then *modify* its W-value next time round in this state.

W-learning [Humphrys, 1996] is a form of dynamic W where W is modified based on (i) whether we were obeyed or not, and (ii) what the new state y is as a result. This can clearly be implemented as an AS_s server. In the pure form of W-learning the Minds do not even share the same suite of actions, and so, for example, cannot simply get together and *negotiate* to find the optimum action. The inspiration was simply to see if competition could be resolved between Minds that had as little in common as possible. That work was unable to give convincing examples where this might arise. Now with the WWM, we see this is the *kind* of model we need when parts of the mind have great difficulty understanding each other (e.g. are written by different remote authors).

3.8 Global Action Selection decisions

If Minds *do* share the same suite of actions, then we can make various global decisions. Say we have `n` Mind servers. Mind server `i`'s preferred action is a_i. It can quantify "how good" action `a` is in state `x` by returning: $Q_i(x,a)$, and can quantify "how bad" action `a` is in state `x` by returning: $Q_i(x,a_i) - Q_i(x,a)$. Then we have 4 basic approaches [Humphrys, 1997]:

1. **Maximize the Best Happiness:**

$$\max_a \max_i Q_i(x,a)$$

   which is equal to static W=Q above, and can be implemented as an AS_s server, with just one query to each Mind server to get its best action and Q-value.

2. **Minimize the Worst Unhappiness:**

$$\min_a \max_i (Q_i(x,a_i) - Q_i(x,a))$$

   which is an AS_m server, requiring multiple queries of each Mind server.

3. **Minimize Collective Unhappiness:**

$$\min_a \left[\sum_i (Q_i(x,a_i) - Q_i(x,a)) \right]$$

   which is an AS_m server.

4. **Maximize Collective Happiness:**

$$\max_a \left[\sum_i Q_i(x,a) \right]$$

   which is an AS_m server.

A number of authors [Aylett, 1995, Tyrrell, 1993, Whitehead et al., 1993, Karlsson, 1997, Ono et al., 1996] implement, using a variety of notations, one of the 4 basic AS methods defined above.

3.9 Nested Mind servers

Digney [Digney, 1996, Digney, 1998] defines Nested Q-learning, where *each* Mind in a collection is able to call on any of the others. Each Mind server has its own set of actions $Q_i(x,a)$ *and* a set of actions $Q_i(x,k)$ where action `k` means "do whatever server `k` wants to do" (as in Hierarchical Q-learning). In a WWM implementation, each Nested server has a list of Mind URLs, either hard-coded or passed to it at startup. So the Nested server *looks* like a $Mind_{AS}$ server co-ordinating many Mind servers to make its decision. But of course it is *not* making the final decision. It is merely *suggesting* an action to the master $Mind_{AS}$ server that coordinates the competition between the Nested servers themselves.

Some of the Nested servers might actually be outside the Action Selection competition, and simply *wait* to be called by a server that *is* in the competition. [Humphrys, 1997] calls these "passive" servers. We have the same with hand-coded $Mind_M$ servers, where some Mind servers may have to wait to be called by others. A server may be "passive" in one Society and at the same time "active" (i.e. the server is in the Action Selection loop) in a different Society.

3.10 Feudal Mind servers

Watkins [Watkins, 1989] defines a *Feudal* (or "slave") Q-learner as one that accepts commands of the form "Take me to state `c`". In Watkins' system, the command is part of the current state. Using the notation `(x,c),a -> (y,c)` the slave will receive rewards for transitions of the form: `(*,c),a -> (c,c)` So the master server drives the slave server by *explicitly* altering the state for it. We do not have to change our definition of the server. It is just that the server driving it is constructing the state `x` rather than simply passing it on from above.

3.11 The sub-symbolic Society of Mind

The Nested and Feudal models are combined in [Humphrys, 1997, Fig. 18.4] showing the general form of a Society of Mind based on Reinforcement Learning. Indeed, the whole model of a complex, overlapping, competing, duplicated, sub-symbolic Society of Mind here is based on the generalised form of a Society of Mind based on Reinforcement Learning.

3.12 Multiple symbolic Mind servers

So far we have only defined a protocol for conflict resolution using numeric weights. Higher-bandwidth communication leads us into the field of *Agents* and its problems with defining *agent communication languages* (formerly symbolic AI knowledge-sharing protocols) that we discussed above. We imagine that numeric weights will be more easily generated by *sub-symbolic* Minds, and are harder to generate in symbolic Minds. This is because symbolic Minds often know *what* they want to do but not "how much" they want to do it. Sub-symbolic Minds, who prefer certain actions precisely because numbers for that action have risen higher than numbers for other actions, may be able to say precisely "how much" they want to do something, and quantify how bad alternative actions would be.

It may be that in the symbolic domain we will make a lot more use of hand-coded $Mind_M$ servers instead of having Minds generating weights to resolve competition. The drawback, of course, is that the $Mind_M$ server needs a lot of intelligence. It needs to understand the goals of all the Mind servers. This relates to the "homunculus" problem, or the need for an intelligent headquarters. Another possibility is the subsidiary Mind servers can be symbolic, while the master $Mind_{AS}$ server is sub-symbolic - e.g. a Hierarchical Q-learner.

3.13 How about robots?

Any scheme of remote re-use is clearly more adapted to virtual agents than real (embodied) ones. But, as discussed above, it is interesting to consider the current lack of direct access to other researchers' robotic experiments. If software animat experiments are hard to replicate, robotic ones are doubly so.

"Telerobotics" is the ability to control a robot remotely. Telerobotic systems have in fact been used in animats [Wilson and Neal, 2000], though not on the network. Outside of the animats field there are in fact a number of "Internet telerobotics" robots that can be controlled remotely over the Internet. [Taylor and Dalton, 1997] discuss some of the issues:

1. We may want a scheme where only one client can control the robot at a time. Whereas with a software-only world one can always allow multiple clients (e.g. by creating a new *instance* of the world for each).
2. The robot owner may want to restrict *who* is able to run a mind on his machine, since some control programs may cause damage.
3. A virtual world server requires little or no maintenance. The author can put the virtual world up on the server and then forget about it. A robotic world server, however, demands much more of a commitment. As a result most of the Internet robots so far have run for a limited time only.

For example, [Stein, 1998] allows remote control of the robot until the client gives it up, or until a timeout has passed. [Paulos and Canny, 1996] operate in a special type of problem space where each action represents the completion of an entire goal, and so actions of different clients can be interleaved. In the robotic tele-garden [Goldberg et al., 1996] users could submit discrete requests at any time, which were executed later by the robot according to its own scheduling algorithm. The robotic Ouija board [Goldberg et al., 2000] is a special type of problem where the actions of multiple clients can be *aggregated*. It seems that all of these schemes could be implemented under the model discussed here. The focus so far in Internet telerobotics has been on remote human-driven control rather than remote *program*-driven control, but this may change.

3.14 How about speed? (robotic worlds and real-time virtual worlds)

Obviously there are inherent performance problems in any system of *remote* re-use. These problems may not be so acute in virtual worlds, where the world can freeze and *wait* for the agent to make a decision. We can *speed up* and *slow down* time in a virtual world to allow for delays caused by the network, without changing the nature of the problem.

This is not, however, an option in *real-time* virtual worlds, such as ones where other users or agents are changing the environment. Here the system may share some of the features of real-time *multi-player online games* (see survey in [Smed et al., 2001]). A large, nested Society of Mind may resemble a peer-to-peer game with low-bandwidth communication, which should scale well. A possible bottleneck is the top-level Mind server, depending how it is designed. [Abdelkhalek et al., 2001] considers performance issues with centralised game servers.

A top-level Mind server is unavoidable because the diversity of suggested actions must be reduced at *some* point, and a decision made. This point is the potential bottleneck. In many of the Action Selection schemes above, the top-level mind is reduced more or less to a *router* rather than a processor in its own right, in an effort to decentralise the intelligence. We now see that such an approach may also be useful in distributing the network *load*.

In the real physical world, a *robotic* animat also needs to make decisions quickly. It may be that a system such as this will be used for *prototyping* - experimenting with different Mind server combinations out of the choices online. Once a combination is chosen, one attempts to get a local installation of all the Mind servers involved. *Why* we are trying to avoid local installation is considered below. If we reject local installation, we cannot avoid network delays.

3.15 What if the remote server is down?

One problem with scaling up AI is that researchers do not want to be dependent on other people's work. What if the remote server is down? Or removed permanently?

Part of the problem is, we argue, models of mind in which the loss of a single server *would* be a serious issue. Instead of models of mind where hundreds of similar servers compete to do the same job, researchers have been assuming the use of *parsimonious* minds where each component does a particular task that is not done by others. A better strategy is to keep adding "unnecessary" duplicated minds to your society. The master $Mind_{AS}$ server asks all Mind servers to suggest actions, and times-out if it does not receive an answer in a short time. So in a highly-duplicated model, if the action does not arrive from one Mind server, it will have arrived from another similar one. In a mind with enough duplication, the temporary network failure (or even permanent deletion) of servers may never even be noticed. Obviously, *some* servers will be essential - like the World server, for instance. The basic answer for how to cope with essential servers is that if it is important to us, we will copy it (if it is free) or buy it or rent it.

[Humphrys, 1997] describes a multiple-minds model of AI that can survive brain damage by re-organising. The reader might have wondered what is the point of that. After all, if the AI is damaged, you just fix it or reinstall it surely? Here is the point - a model of AI that can survive *broken links*.

3.16 How about multi-agent systems?

In distributed intelligence, there are two major camps:

1. Multiple minds in one body, competing for expression (the fields of action selection, motivation, goal conflict, emotion, Society of Mind). We will refer to this as the "AS" camp.

2. Multiple bodies, which can act independently (multi-agent systems, collective behaviour). We will refer to this as the "MAS" camp.

The field of Animats focuses on both AS and MAS *offline*. The field of Internet Agents focuses on MAS online. The WWM focuses on AS online - on addressing the issue of how to construct really complex agent minds, and implementing action selection across servers. This, we argue, is the neglected area.

There is perhaps one other neglected area, which is *sub-symbolic* MAS online. Most work on MAS online is at the symbolic level (see *agent communication languages*, as discussed previously). One interesting issue is whether the Animats work on MAS, which involves what might be called *sub-symbolic* communication or signalling, can be brought online. Ongoing research by Walshe [Walshe, 2001] will attempt to interface sub-symbolic AS and MAS online.

4 Implementation of the WWM

We now describe the first implementation of the WWM, and the decisions made in reaching that implementation.

4.1 Rejecting local installation

First note that we reject the solution that would have been imagined for most of the history of AI - local installation. Given the huge diversity of, and incompatibility of, operating systems, platforms, files, libraries, versions, environments, programming methodologies and programming languages in use in AI (a diversity perhaps actually greater in AI than in any other field of computing), we view it as highly unlikely that local installation could lead to widespread re-use. We clearly reject the idea of asking all animats researchers to use a certain programming language (e.g. Java) or platform.

How can one avoid these compatibility problems and allow researchers use whatever platform they want? By *server-side* programs rather than client-side programs. The Web demonstrates this highly successful model of re-use - *leaving* the program on the remote Web server, and running it from there. One strange aspect of adopting this model for the WWM is that the mind may consist of components which are physically at different remote sites, and which stay there, and just communicate with each other remotely. Hence the mind is *literally* decentralised across the world - something which has never existed in nature. Hence the name, the "World-Wide-Mind".

4.2 Rejecting models designed for network programmers

Given that we propose a remote solution, we might look at the emerging *web services* architectures for program-to-program transactions online. These are called "web services" because they run over the existing HTTP network (rather than demand a new network be set up). The emerging standard is to send messages to remote objects or programs using the SOAP message system [w3.org/TR/SOAP], which runs over normal HTTP. SOAP messages are based on XML [w3.org/XML], a standard meta language used to describe data in tagged plaintext (i.e. it looks similar to HTML).

While we agree with this general scheme (run on HTTP, the data format should be tagged and extensible), we reject using the full complexity of the web services protocols. Why? Because of the unique nature of our audience. Most animats researchers are programmers, but not network programmers. These protocols - and indeed almost *all* protocols in computer networks, web services, distributed objects or Internet Agents - assume the programmer is a networks specialist (or is willing to become one). SOAP messages are complex, and you require an API to parse them. Doing it yourself is difficult.

4.3 Rejecting unforgiving data formats

We also reject strict XML. XML has moved away from the *forgiving* nature of HTML. In XML, opening and closing tags must both be present, the tags in a document must form a *tree*, and so on. Any failure results in the document being rejected by the parser. XML parsers are also extremely complex to use.

We still agree with the idea of *tagged plain text* for our data. Plain text is important so humans can read the data, and programmers can parse it themselves. With tagged plain text (each piece of data is delimited by tags, whitespace is ignored) it is much easier to create a *tolerant* parser than with untagged plain text (where, say, *precise* column number or line number defines which piece of data is which). Tagging allows *extensible* systems - we can ignore new tags that we don't recognise.

4.4 Lightweight Web Services

We describe our approach as *Lightweight* Web Services [O'Leary, 2002a]. These are web services that anyone can create without having to learn a whole family of new protocols. It should be (almost) as easy to create a Lightweight Web Service as it is to create a Web Page. Under the system we have now developed, any animats researcher can put their animat "mind" or "world" online as a WWM server by converting it into a command-line program that reads standard input and writes to standard output. The program can be written in *any* language and runs as a "CGI script". The input and output is in a stripped-down, forgiving, XML-like language called AIML. We now explain this.

4.5 CGI

All Web servers support a system of server-side programs called CGI. CGI is *not* a difficult technology - indeed, there is almost nothing to it except placing a command-line program in the CGI directory of your Web server. Any programming language may be used. Programs read plaintext input (text, HTML, XML, or any XML-like format) on standard input and write plain text output to standard output. All browsers (and other clients) can run remote CGI programs.

CGI is the command-line of the Internet. Network enthusiasts often neglect CGI and describe *much* more complex technologies (JavaScript, VBScript, Java applets, Java Servlets, ASP, JSP, etc.) we suspect precisely *because* CGI is so simple and was worked out long ago (1993, before the Web took off). But CGI is not obsolete. CGI is still a *far* simpler technology than these or *any* other technology for programs online either at client-side or server-side.

4.6 AIML

The plaintext input to and output from the WWM servers is in a simple, loosely-defined XML-like language we call AI Markup Language (AIML). Strictly speaking, AIML is not XML since we reject strict XML formatting. AIML is closer to HTML in that we try never to reject messages because of their being badly formed. A best effort is always attempted. We will *have* to be tolerant of loosely-defined servers on the WWM because often there will be *no alternative* to the AI author. If Bloggs does not put the Bloggs learning algorithm online, it will often be the case that *no one else will.* So we *can't* just refuse to use his server if it generates sloppy AIML.

This tolerance does not mean we cannot issue *recommendations.* The situation will be like the Web. The portal site `w3.org` defines the official HTML spec. (e.g. "tables should end with an end-table tag"). But the browser can't just choke on bad HTML, not if there is scope to make a guess and display it (e.g. if end of file comes with no end-table tag, then insert end-table tag). The browser *must* tolerate bad HTML, or users will switch to browsers that do. And the pool of authors would never have grown so big if authors had to write strict HTML. It is often forgotten that the Web *does not* run on strict HTML, and never could have.

Similarly, the portal site `w2mind.org` will define an official AIML spec. (e.g. "WWM query responses should end with an end-response tag"). But no matter how we define it, there will always be room for the client to make some guesses with bad AIML. Clients must try to tolerate AIML "close to" the spec. - though obviously there can be no guarantees once one deviates from the spec.

4.7 How to write a WWM server

Now we bring this all together. To write a Mind server that can suggest an action given that the world is in some state `x`, one writes a command-line program that can parse something like this on standard input:

```
<request type="GetAction" runid="RUNID">
    <data name="x"> x </data>
</request>
```

where the format of x is decided by the World server. Clearly, this plaintext input can easily be parsed by any programmer using simple string searching mechanisms in any language (the first author's parser is just 5 lines of UNIX Shell, and is tolerant of many different variations in the input AIML). The Mind server then outputs to standard output something like:

```
<response type="GetAction" runid="RUNID">
    <data name="a"> a </data>
</response>
```

where the format of a is decided by the World server. We say "something like" because AIML is still in a state of revision. An agreed standard will be released in mid-2002. For the current draft, AIML v1.1, see [O'Leary, 2002].

That is all one needs - to agree on the format of AIML - and even full agreement is not necessary if one writes a tolerant parser. The program can be written in any language. Input and output can be debugged using an ordinary web browser (though for *repeated* queries one would want to use one of our dedicated clients).

4.8 Existing work

To date, we have put online:

1. World servers representing simple toroidal "grid worlds" with "food" and mobile "predators", written in C++ and Java by multiple authors.
2. Mind servers to drive the animat in these worlds, written in C++ and Java by multiple authors.

The servers are hosted on *separate* remote Web (HTTP) servers. Both GUI and command-line clients have been written for Windows and UNIX. Using the clients, Mind A (Java) was able to explore World B (C++), and Mind B (C++) was able to explore World A (Java).

4.9 Further issues

There are many further implementation issues, such as:

1. How to save the *state* of the world (or mind, if it is learning) in between requests.
2. How to write CGI programs that are *persistent* in memory between requests.
3. How to handle multiple users.
4. How to display graphically what is happening in the world (or inside the mind) and where to display this.
5. How does a server call another server.

We deal with all of these issues in [Humphrys, 2001, O'Leary, 2002a]. But it remains that a basic WWM server can still be got running with just a program that parses plaintext and outputs plaintext as above.

5 Future work

From the WWM viewpoint the next immediate thing to do is finalise a standard for AIML and then publicly release this standard, plus servers and clients that use it, and make these available from the portal site `w2mind.org`. Then other researchers can start building their own servers.

From the *animats* viewpoint the next things to do are: (a) Put existing well-known minds and worlds online as servers (we already have Tyrrell's world running [Tyrrell, 1993] but not yet as a server) and: (b) Construct *network action selection* mechanisms for Action Selection across multiple *remote* minds by different authors.

6 Conclusion

This paper has argued for the need to decentralise the work in AI so that researchers may specialise on different parts, *and* a mind may be constructed from these multiple specialist parts. Such a future (of specialists coming together) has been imagined (at least implicitly) in many branches of AI, but no practical scheme for implementing it has yet emerged. We believe that now, with server-side programming ubiquitous on the Internet, such a scheme is finally possible.

We have a new vision of a mind: no *single* author could write a high-level artificial mind, but perhaps the *entire* scientific community could. Each *piece* will be understood by someone, but the whole may be understood by no-one. Perhaps we need a new respect for the *magnitude* of the AI problem - that building a high-level artificial mind may be on the same scale as constructing something like a national *economy*, or the city of London. No single individual or company built London or New York. But humanity as a whole did.

Acknowledgements

The software for this system (servers, clients and server support software) and the design of AIML is the joint work of the authors of this paper and Dave O'Connor and Ray Walshe. We are grateful to two anonymous referees for their comments.

References

Abdelkhalek, A.; Bilas, A. and Moshovos, A. (2001), Behavior and Performance of Interactive Multiplayer Game Servers, *Proc. Int. IEEE Symposium on the Performance Analysis of Systems and Software.*

Aylett, R. (1995), Multi-Agent Planning: Modelling Execution Agents, *14th UK Planning and Scheduling SIG.*

Berger, H.W. (1998), *Is The NP Problem Solved? (Quantum and DNA Computers)*, www.pcs.cnu.edu/~hberger/Quantum_Computing.html

Brooks, R.A. (1986), A robust layered control system for a mobile robot, *IEEE Journal of Robotics and Automation* 2:14-23.

Brooks, R.A. (1991), Intelligence without Representation, *Artificial Intelligence* 47:139-160.

Brooks, R.A. (1997), From Earwigs to Humans, *Robotics and Autonomous Systems*, Vol. 20, pp. 291-304.

Brooks, R.A. et al. (1998), The Cog Project, *Computation for Metaphors, Analogy and Agents*, Springer-Verlag.

Bryson, J. (2000), Cross-Paradigm Analysis of Autonomous Agent Architecture, *JETAI* 12(2):165-89.

Daniels, M. (1999), Integrating Simulation Technologies With Swarm, *Workshop on Agent Simulation*, Univ. Chicago, Oct 1999.

de Garis, H. (1996), CAM-BRAIN: The Evolutionary Engineering of a Billion Neuron Artificial Brain, *Towards Evolvable Hardware*, Springer.

Dennett, D.C. (1978), Why not the whole iguana?, *Behavioral and Brain Sciences* 1:103-104.

Digney, B.L. (1996), Emergent Hierarchical Control Structures, *SAB-96.*

Digney, B.L. (1998), Learning Hierarchical Control Structures for Multiple Tasks and Changing Environments, *SAB-98.*

Ginsberg, M.L. (1991), Knowledge Interchange Format: The KIF of Death, *AI Magazine*, Vol.5, No.63, 1991.

Goldberg, K. et al. (1996), A Tele-Robotic Garden on the World Wide Web, *SPIE Robotics and Machine Perception Newsletter*, 5(1), March 1996.

Goldberg, K. et al. (2000), Collaborative Teleoperation via the Internet, *IEEE Int. Conf. on Robotics and Automation (ICRA-00).*

Guillot, A. and Meyer, J.-A. (2000), From SAB94 to SAB2000: What's New, Animat?, *SAB-00.*

Harvey, I.; Husbands, P. and Cliff, D. (1992), Issues in Evolutionary Robotics, *SAB-92.*

Humphrys, M. (1996), Action Selection methods using Reinforcement Learning, *SAB-96.*

Humphrys, M. (1997), *Action Selection methods using Reinforcement Learning*, PhD thesis, University of Cambridge, Computer Laboratory. `www.compapp.dcu.ie/~humphrys/PhD`

Humphrys, M. (2001), *The World-Wide-Mind: Draft Proposal*, Dublin City University, School of Computer Applications, Technical Report CA-0301, February 2001. `www.compapp.dcu.ie/~humphrys/WWM`

Humphrys, M. (2001a), Distributing a Mind on the Internet: The World-Wide-Mind, *ECAL-01*, Springer-Verlag LNCS/LNAI 2159, September 2001.

Kaelbling, L.P.; Littman, M.L. and Moore, A.W. (1996), Reinforcement Learning: A Survey, *JAIR* 4:237-285.

Karlsson, J. (1997), *Learning to Solve Multiple Goals*, PhD thesis, University of Rochester, Department of Computer Science.

Lin, L-J (1993), Scaling up Reinforcement Learning for robot control, *10th Int. Conf. on Machine Learning.*

Martin, F.J.; Plaza, E. and Rodriguez-Aguilar, J.A. (2000), An Infrastructure for Agent-Based Systems: an Interagent Approach, *Int. Journal of Intelligent Systems* 15(3):217-240.

McDermott, D. (1997), "How Intelligent is Deep Blue?", *New York Times*, May 14, 1997.

Minsky, M. (1986), *The Society of Mind.*

Minsky, M. (1991), Society of Mind: a response to four reviews, *Artificial Intelligence* 48:371-96.

Nilsson, N.J. (1995), Eye on the Prize, *AI Magazine* 16(2):9-17, Summer 1995.

O'Leary, C. (2002), *AIML v1.1 - Artificial Intelligence Markup Language*, `comp.dit.ie/coleary/research/phd/wwm/`

O'Leary, C. (2002a), *Lightweight Web Services for AI Researchers*, `comp.dit.ie/coleary/research/phd/wwm/`

Ono, N.; Fukumoto, K. and Ikeda, O. (1996), Collective Behavior by Modular Reinforcement-Learning Animats, *SAB-96.*

Paulos, E. and Canny, J. (1996), Delivering Real Reality to the World Wide Web via Telerobotics, *IEEE Int. Conf. on Robotics and Automation (ICRA-96).*

Sloman, A. and Logan, B. (1999), Building cognitively rich agents using the SIM_AGENT toolkit, *Communications of the ACM*, 43(2):71-7, March 1999.

Smed, J.; Kaukoranta, T. and Hakonen, H. (2001), Aspects of Networking in Multiplayer Computer Games, *Proc. Int. Conf. on Application and Development of Computer Games in the 21st Century.*

Stein, M.R. (1998), Painting on the World Wide Web, *IEEE / RSJ Int. Conf. on Intelligent Robotic Systems.*

Sutton, R.S. and Santamaria, J.C., A Standard Interface for Reinforcement Learning Software, www-anw.cs.umass.edu/~rich/RLinterface/RLinterface.html

Taylor, K. and Dalton, B. (1997), Issues in Internet Telerobotics, *Int. Conf. on Field and Service Robotics.*

Tyrrell, T. (1993), *Computational Mechanisms for Action Selection*, PhD thesis, University of Edinburgh.

Walshe, R. (2001), The Origin of the Speeches: language evolution through collaborative reinforcement learning, *Proc. 3rd Int. Workshop on Intelligent Virtual Agents (IVA-2001).*

Watkins, C.J.C.H. (1989), *Learning from delayed rewards*, PhD thesis, University of Cambridge.

Whitehead, S.; Karlsson, J. and Tenenberg, J. (1993), Learning Multiple Goal Behavior via Task Decomposition and Dynamic Policy Merging, *Robot Learning*, Kluwer.

Wilson, M. and Neal, M. (2000), Telerobotic Sheepdogs: How useful is autonomous behavior?, *SAB-00.*

Wilson, S.W. (1990), The animat path to AI, *SAB-90.*